Einführung in die Programmierung

WS 2016/2017, Blatt 5

Bauhaus-Universität Weimar

Prof. Norbert Siegmund
Nathalie Dittrich, Hans Lienhop

Ausgabe: 06.01.2017
Abgabetermin: Dienstag,17.01.2017, 12:00 Uhr

Abgabe bitte an folgende eMail schicken: hans.lienhop@uni-weimar.de

Aufgabe 1 Heap (3 Punkte)

Führen Sie den folgenden Java-Code aus. Analysieren Sie die Fehlermeldung und finden Sie Möglichkeiten, diese zu vermeiden.

```
class Example{
        static final int SIZE=1000*1024*1024;

        public static void main(String[] a) {
          int[] i = new int[SIZE];
         }
      }
```

Aufgabe 2 Pascalsches Dreieck (7 Punkte)

Informieren Sie sich über den Aufbau des Pascalschen Dreiecks. Implementieren Sie ein Programm, welches ein *Integer* entgegen nimmt und das Pascalsche Dreieck bis zu dieser Zeile rekursiv berechnet und ausgibt. Das Dreieck muss bei der Ausgabe nicht zentriert sein.

Aufgabe 3 Währungsrechner (3 Punkte)

Implementieren Sie eine Klasse, welche Geldbeträge in andere Währungen umrechnen kann. Dabei soll eine Kombination aus *enum* und *switch − case* verwendet werden.

Aufgabe 4 Doppelt verkettete Liste (30 Punkte)

Erstellen Sie einen Datentyp (eine Klasse `DLList`) für doppelt verkettete Listen. In der Liste sollen Strings gespeichert werden. Die Knoten sollen eine Referenz auf den nachfolgenden als auch auf den vorhergehenden Knoten haben.

(a) Die Knoten Klasse (`Node`) soll folgende Eigenschaften implementieren:

(i) `public Node(String data)` - der Konstruktor.

(ii) `public String toString()` soll {*stringvalue*} ausgeben.

(iii) `public void setNextNode(Node next)` setzt den nächsten Knoten auf `next`.

(iv) `public void setPreviousNode(Node previous)` setzt den vorherigen Knoten auf `previous`.

(v) `public Node getNextNode()` gibt die Referenz auf den nächsten Knoten oder `null` zurück, falls der Knoten der letzte der Liste ist.

(vi) `public Node getPreviousNode()` gibt die Referenz auf den vorherigen Knoten oder `null` zurück, falls der Knoten der erste der Liste ist.

(vii) `public String getValue()` gibt den gespeicherten Wert zurück.

(b) Folgende Methoden sollen von der Listen Klasse implementiert werden:

(i) `public boolean isEmpty()` gibt zurück ob die Liste leer ist.

(ii) `public void display()` gibt die Liste auf der Konsole aus.

(iii) `public void add(int pos, String content)` fügt den `String content` an der Position `int pos` ein.

(iv) `public void add(String content)` fügt `String content` am Ende der Liste ein.

(v) `public void remove(String content)` löscht den ersten Knoten, der `String content` enthält aus der Liste.

(vi) `public void removeFirst()` löscht den ersten Knoten der Liste.

(vii) `public void removeLast()` löscht den letzten Knoten der Liste.

(viii) `public void clear()` löscht alle Knoten aus der Liste.

(ix) `public String getFirst()` gibt den String des ersten Knotens der Liste zurück.

(x) `public String getLast()` gibt den String zurück, der im letzten Knoten der Liste gespeichert ist.

(xi) `public String get(int pos)` gibt den String an der Position `int pos` zurück.

(xii) `public void concat(DLList list)` fügt `DLList list` am Ende der Liste an.

(xiii) `public int find(String content)` überprüft ob der `String content` in der Liste enthalten ist. Ist dies der Fall, so gibt die Methode die Knotennummer zurück oder -1 falls der String nicht enthalten ist.

(xiv) `public boolean contains(String content)` gibt zurück ob die Liste `String content` enthält.

(xv) `public int size()` gibt die Anzahl der Knoten zurück.

Testen Sie Ihre Implementierung ausführlich!